WHAT IS CLAIMED IS:

1. In a computer-system, a method comprising:

receiving a request via a process thread having a first memory map associated therewith;

changing a privilege level to a level that allows a map change;

performing the map change to associate a second memory map with the process thread, the second memory map providing different memory access with respect to the first memory map; and

restoring the privilege level to a level that does not allow a map change.

2. The method of claim 1 wherein receiving a request comprises receiving an application programming interface call at an operating system component.

3. The method of claim 1 wherein receiving a request comprises, receiving at an operating system a call from a kernel mode component.

4. The method of claim 3 wherein the kernel mode component comprises an installable driver.

5. The method of claim 1 wherein changing a privilege level comprises calling a call gate.

6. The method of claim 1 wherein changing a privilege level comprises changing to a ring 0 privilege level.

7. The method of claim 1 wherein performing the map change comprises writing to a register.

8. The method of claim 1 wherein the second memory map accesses protected memory, and further comprising, executing trusted code while the second memory map is associated with the process thread.

9. The method of claim 8 further comprising, performing a second map change to re-associate the first map with the process thread.

10. The method of claim 8, wherein executing trusted code includes entering a function at a predefined entry point.

11. The method of claim 10 wherein entering the function comprises making an application programming interface call.

12. The method of claim 10 wherein the function allocates memory.

13. The method of claim 10 wherein the function deallocates memory.

14. The method of claim 10 wherein the function allocates an object.

15. The method of claim 14 wherein the object comprises a handle.

16. The method of claim 14 wherein the object comprises a synchronization objects.

17. The method of claim 14 wherein the object comprises a process.

18. The method of claim 14 wherein the object comprises a threads.

19. The method of claim 10 wherein the function performs a trust-privileged operation.

20. The method of claim 19 wherein the trust-privileged operation comprises signaling a synchronization object.

21. The method of claim 19 wherein the trust-privileged operation comprises deleting a timer.

22. The method of claim 19 wherein the trust-privileged operation comprises closing a handle.

23. The method of claim 1 wherein the first and second memory maps each include a mapping that maps a virtual memory address to a physical memory address that is larger than the largest possible virtual memory address that an entity is allowed to address.

24. The method of claim 23 wherein the virtual memory address that maps to a physical memory address that is larger is in user mode addressable space.

25. The method of claim 23 wherein the first and second memory maps each include a mapping that maps a virtual memory address to a physical memory address that is the same.

26. The method of claim 25 wherein the physical memory address that is the same is in kernel mode addressable space.

27. The method of claim 23 wherein the first and second memory maps each include a mapping that maps a virtual memory address to a physical memory address that is the same, wherein the virtual memory address that maps to a physical memory address that is larger is in user mode addressable space, and wherein the physical memory address that is the same is in kernel mode addressable space.

28. The method of claim 1 wherein the first and second memory maps each map a virtual memory address to a physical memory address that is common to both maps.

29. The method of claim 1 wherein the second map maps to memory that is invalid in the first map.

30. The method of claim 1 wherein the second map maps to memory that has different access rights in the first map.

31. A computer-readable medium having computer-executable instructions for performing the method of claim 1.

32. In a computing device, a system comprising:

a process having at least one thread;

a first memory map associated with the thread and having data therein that maps virtual memory addresses to physical memory;

a second memory map having data therein that maps virtual memory addresses to physical memory, the second memory map providing different memory access with respect to the first memory map;

a protection mechanism, the protection mechanism configured to allow changing of a map; and

trusted code, the trusted code configured to invoke the protection mechanism to change the thread from being associated with the first map to be being associated with the second map.

33. The system of claim 32 wherein the second memory map has more access rights to virtual memory addresses than the first memory map.

34. The system of claim 32 wherein the protection mechanism comprises a call gate configured to change privilege levels.

35. The system of claim 32 wherein the trusted code includes a thunk configured to re-vector a function call directed to one set of code to another set of code.

36. The system of claim 32 wherein the trusted code further includes a function.

37. The system of claim 36 wherein the function allocates memory to the process.

38. The system of claim 36 wherein the function deallocates memory.

39. The system of claim 36 wherein the function allocates an object.

40. The system of claim 39 wherein the object comprises a handle.

41. The system of claim 39 wherein the object comprises a synchronization objects.

42. The system of claim 39 wherein the object comprises a process.

43. The system of claim 39 wherein the object comprises a threads.

44. The system of claim 36 wherein the function performs a trust-privileged operation.

45. The system of claim 44 wherein the trust-privileged operation comprises signaling a synchronization object.

46. The system of claim 44 wherein the trust-privileged operation comprises deleting a timer.

47. The system of claim 44 wherein the trust-privileged operation comprises closing a handle.

48. The system of claim 32 wherein only the trusted code is executed while the second memory map is in use.

49. The system of claim 32 wherein the trusted code executes in response to a call from the process.

50. The system of claim 49 wherein the trusted code comprises an operating system component, and wherein the

trusted code executes in response to an application programming interface call from the process to an operating system component.

51. The system of claim 32 wherein the protection mechanism comprises a call gate.

52. The system of claim 32 wherein the trusted code changes the thread from being associated with the first map to be being associated with the second map by writing to a register.

53. The system of claim 32 wherein the trusted code changes the thread from being associated with the first map to be being associated with the second map by instructing a hardware component to select a different subset of a translation look-aside buffer.

54. The system of claim 32 wherein the trusted code performs a second map change to re-associate the first map with the process thread, and invokes the protection mechanism to not allow map changing.

55. The system of claim 54 wherein the protection mechanism changes a privilege level to not allow map changing.

56. The system of claim 32 wherein the first and second memory maps each include a mapping that maps a virtual memory address to a physical memory address that is larger than the largest possible virtual memory address that an entity is allowed to specify.

57. The system of claim 56 wherein the virtual memory address that maps to a physical memory address that is larger is in user mode addressable space.

58. The system of claim 56 wherein the first and second memory maps each include a mapping that maps a virtual memory address to a physical memory address that is the same.

59. The system of claim 56 wherein the physical memory address that is the same is in kernel mode addressable space.

60. The system of claim 32 wherein the first and second memory maps each map a virtual memory address to a physical memory address that is common to both maps.

61. The system of claim 32 wherein the second map maps to memory that is invalid in the first map.

62. The system of claim 32 wherein the second map maps to memory that has different access rights in the first map.

63. The system of claim 32 wherein the second map shares a mapping of some virtual addresses to physical addresses common to the first map, and includes another mapping of virtual addresses to physical addresses that are not common to the first map.

64. A computer-implemented method, comprising:

associating first and second address maps with a process, wherein at least the second address map includes a mapping that maps a virtual address to a physical address that is larger than the largest possible virtual memory address;

receiving a request from a thread of the process to change from the first address map to the second address map;

changing the first address map to the second address map; and

using the mapping to access data at a physical memory location having a physical address that is larger than the largest possible virtual memory address.

65. The computer-implemented method of claim 64 wherein the first and second memory maps each map a virtual memory address to a physical memory address that is the same.

66. The computer-implemented method of claim 65 wherein each virtual memory address that maps to a physical memory address that is larger is in user mode addressable space, and wherein the physical memory address that is the same is in kernel mode addressable space.

67. The computer-implemented method of claim 64 further comprising a third map having a mapping that maps a virtual address to a physical address that is larger than the largest physical address mapped to by the second map, and further comprising switching to the third map to access data at the physical address that is larger than the largest physical address mapped to by the second map.

68. The computer-implemented method of claim 64 wherein changing the first map to the second map includes calling the operating system to switch the maps.

69. A computer-readable medium having computer-executable instructions for performing the method of claim 64.

70. A computer-implemented method, comprising:

associating first and second address maps with a process, wherein the second address map provides different memory access with respect to the first memory map;

running trusted code with the first map;

switching to the second map prior to running a first set of untrusted code without switching the process; and

returning to the first map after completion of the untrusted code.

71. The computer-implemented method of claim 70 wherein switching to the second map includes calling the operating system to switch the maps.

72. The computer-implemented method of claim 70 wherein the first and second maps map to at least one physical address that is the same.

73. The computer-implemented method of claim 70 further comprising switching to a third map prior to running a second set of untrusted code without switching the process.

1033720.82555.650

74. The computer-implemented method of claim 70 wherein the first and third maps map to at least one physical address that is the same.

75. The computer-implemented method of claim 70 wherein the second and third maps map to at least one physical address that is the same.

76. The computer-implemented method of claim 70 wherein each of the maps map to at least one physical address that is the same.

77. A computer-readable medium having computer-executable instructions for performing the method of claim 70.